CSCI 3104: Algorithms
Spring 2019

Grand Challenge

Option 2: Network Motifs

Since the late 1990s, there has been an explosive increase in the number of large data sets taking the form of networks. This includes

- social networks (which had been studied on a smaller scale by sociologists since the 1960s if not earlier),
- biological networks (which genes regulate which, which proteins bind with which),
- food webs (which species eat which other species),
- technological networks (the internet, phone networks),
- transportation networks (airline routes, roads, worldwide shipping networks),
- economic networks (trade, boards of directors, stock markets), and
- informational networks (words that appear near each other, citations in academic papers, citations in legal rulings, web pages that link to each other).

Among these so-called “real-world complex networks”, many common structural features were noticed (or proposed and then long-debated): for example, the small-world phenomenon (“six degrees of separation”), the “hub” phenomenon (most people have few friends, but a small number of people have a *huge* number of friends), and “clustering” (a friend of your friend tends to be your friend). These common features raised the exciting possibility that there might be universal, or at least wide-spread, mechanisms that account for the common structure observed in many of these networks, despite the fact that they come from seemingly unrelated domains.

The question is: what mechanisms lead to the features we see in real-world networks, and how can we leverage these features to learn more about these networks?

Milo *et al.* [1] proposed that the prevalence of different small sub-networks (“network motifs”) in a network could tell us a lot about the larger network’s structure and function. For example, triangles tend to be highly abundant in social networks, because a friend of your friend tends to be your friend as well. They went so far as to propose network motifs as the “building blocks” of complex networks, in the same way that transistors, adders, and other small sub-circuits form the building blocks of a CPU.

The limiting factor in identifying network motifs is counting the occurrences of small subgraphs inside a larger graph. Your project is to develop such a subgraph counting algorithm.

Background

A **graph** or **network** consists of a set of **vertices** or **nodes**, and a collection of **edges** or **links** connecting pairs of vertices. Formally, we write G=(V,E), where V is the vertex set, and E is the edge set. While these terms are interchangeable, we will tend to use “graph” for the small graphs we are searching for, and “network” for the big graph we are searching in.

Each edge in E is specified by a pair (u,v) of vertices. In this project, we will only considered graphs that are **undirected**, meaning that the edge (u,v) is considered to be identical to the edge (v,u). Let e=(u,v) be an edge in a graph G (see Fig. 1). Then we say that u and v are the **endpoints** of e, e is **incident** to u and to v, and u and v are **neighbors** in G**.**

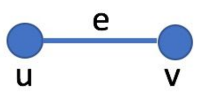

Figure 1: An edge.

A graph is **simple** if it does not contain any self-loops (edges of the form (v,v)) and if between every pair of vertices there is at most one edge. ***We will only consider simple, undirected graphs in this project.***

A **sub-graph** or **sub-network** of a network G=(V,E) is another graph H=(V_H, E_H) such that V_H is a subset of V, and E_H is a subset of E. If H is a subgraph of G, we say H is an **induced subgraph** if E_H is precisely the set of all edges in E which connect the vertices in V_H. Symbolically, H is induced if E_H= {e in E such that both endpoints of e are in V_H}. Given just V_H as a subset of V, we say that $\mathbf{V_H}$ **induces H** (where G is understood from context, or may need to be specified). Note that every subset V’ of V induces a unique subgraph.

For example, in the graph in Fig. 2, the vertices [1,2,3,4] induce a 4-cycle, but the vertices [5,6,7,8] do not (because of the presence of the edge [5,8]).

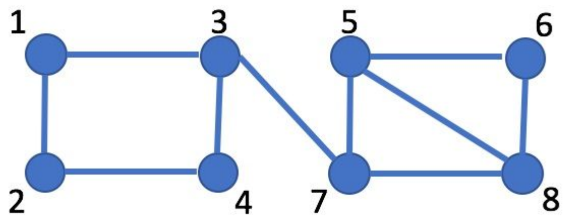

Figure 2: Induced vs. non-induced subgraphs

A **path** in a graph G is a sequence of vertices $(v_1, v_2, ..., v_k)$ such that (v_i, v_{i+1}) is an edge of G for all $1 \leq i < k$. G is **connected** if, for every pair of vertices u,v in G, there is a path from u to v in G. For example, in the graph above is connected, but if we removed the edge [3,7] it would no longer be connected. ***In this project, we will only be interested in connected subgraphs.***

A useful concept is that of isomorphism, which is a notion of when two graphs are "essentially the same." Given two graphs G=(V,E) and G'=(V',E'), a function f:V -> V' is an **isomorphism** from G to G' if

1. f is bijective, that is, it is one-to-one and onto.
2. (u,v) is an edge of G if and only if (f(u), f(v)) is an edge of G'

If there is any isomorphism from G to G' then we say that G and G' are **isomorphic**. Two isomorphic graphs are usually considered "the same", even though they may not be literally the same when represented in a computer. The following two figures give examples of isomorphic and non-isomorphic graphs.

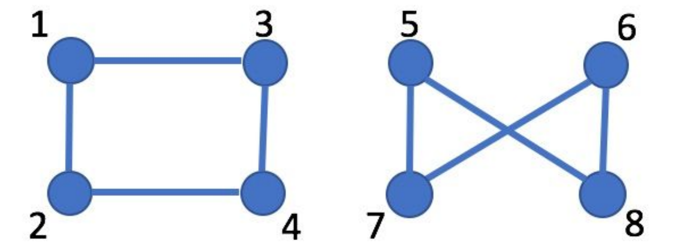

Figure 3: These two graphs are isomorphic.
One isomorphism is given by the map 1->5, 3->8, 4->6, 2->7.

Note that in Figure 3 there is no vertex at the "intersection" of the edges [5,8] and [6,7]. This "intersection" is in fact not part of the graph at all! It just an artifact of how it was drawn.)

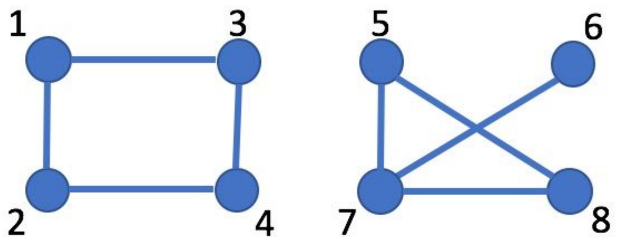

Figure 4: These two graphs are not isomorphic, despite having the same number of vertices (4) and the same number of edges (4). Try to prove they're not isomorphic!

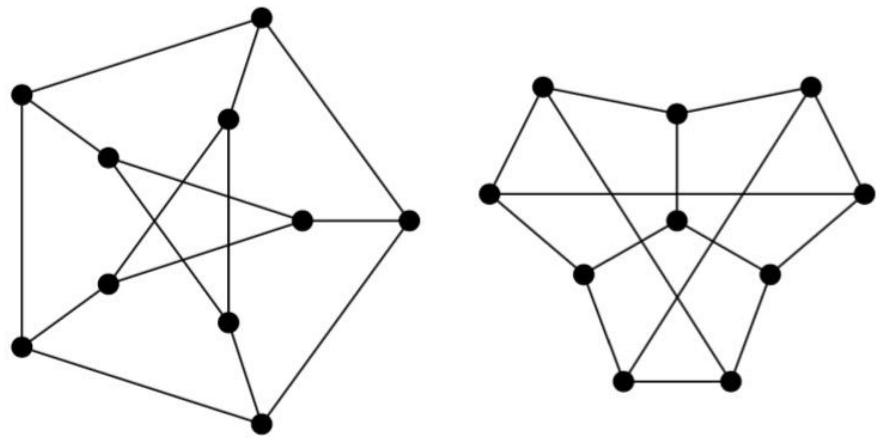

Figure 5: These two graphs are isomorphic, but it's not immediate to see. (Remember that there are only vertices where there are dots, not at other "intersections" of edges.) This is in fact a famous graph called the Petersen graph.

The **isomorphism type** or **isomorphism class** of a graph G is the set of all graphs that are isomorphic to G. When we speak of a list of isomorphism types of graphs, we mean that the list contains exactly one graph from each isomorphism class.

Given a graph H (which we think of as small) and a network data set G (which we think of as relatively large), an **instance** or **occurrence** of H in G is an induced subgraph H' of G such that H and H' are isomorphic. For example, in Figure 2, the vertices [1,2,3,4] form an instance of a 4-cycle, and the vertices [5,7,8] are an instance of a triangle. ***In this project we will only consider induced subgraphs***.

For counting subgraphs in the context of network motifs, there are currently two main standard techniques (though there are others as well):

1. Network-centric: For a given k, count all instances of all subgraphs (up to isomorphism - e.g., all triangles get counted together) of size at most k in G, or
2. Motif-centric: For a given "query graph" H, count all instances of H in G

The second strategy here is technically more general (since if you have a solution to the second strategy, you can use it, along with a list of all isomorphism types of k-node graphs, to implement the first strategy - do you see how?), but both strategies have advantages and disadvantages, and each may lend itself to different kinds of optimization. There are also other possible strategies.

Summary of assumptions:

- All graphs we consider are simple, undirected graphs

- We only consider induced, connected subgraphs

Goal / quality of solution

Given a network data set G, your goal is: for as many and as large a graph H as you can manage, count the number of occurrences of H in G. (Remember, we are only considering induced, connected subgraphs.)

For each network data set G, the "quality" of your solution can be evaluated by two numerical criteria (all solutions must count the correct number of occurrences):

1. What is the largest k such that you are able to find all occurrences of *all connected graphs H* on at most k vertices in under an hour on a standard laptop? Any strategy you employ (network-centric, motif-centric, or otherwise) should be able to perform this operation.
2. [If you implemented a motif-centric algorithm] What is the largest k such that there is some connected graph H on k vertices such that you can find all occurrences of H in under an hour on a standard laptop? **Note: You do not need to implement a motif-centric algorithm to get full credit.** We've included this second criterion because motif-centric algorithms can frequently look for *some* query graphs that are much larger than the k they could reach in answer to criterion 1. So for motif-centric algorithms, it makes sense to use this is a second metric.

Data

We will provide 3 networks of increasing size (and, correspondingly, difficulty). Your team may attempt to count subgraphs in any or all of these networks. Shortly before the project is due, we will release a 4th network.

We will also provide you a list of all (isomorphism classes of) connected graphs on k vertices, for k up to 8 (there are already 11,117 isomorphism classes of simple, undirected, connected graphs on 8 vertices). If you need such a list for larger k, we will attempt to provide it (but for k=9 there are 261,080 of them; see here).

Rules

- No extra parameters. You cannot pass in special parameters for different input networks G.
- No libraries. You must implement all data structures and algorithms from scratch.
- Cite all papers, GitHub repos, websites, books, or other sources that made meaningful contributions to your solution.

What to submit.

1. Well-formatted and well-documented **source code**.
2. **Output subgraph counts.** For each network data set G of those we provide you (up to 4):
    - For each small graph H whose instances you count:
        - If H has at most 8 nodes, then the number of nodes k, and the index of H in the corresponding list of all k-node connected graphs we provide you (0-indexed!), and the number of (induced) occurrences of H in G
            - For example, if you count 110 paths on 3 vertices and 20 triangles in a graph G, your file would contain lines of the form k, index, number, as in:

              ```
              3 0 110
              3 1 20
              ```

              (Assuming that in our list of graphs on 3 vertices, the path on 3 vertices has index 0 and the triangle has index 1.)
        - If H has more than 8 nodes, than a description of H as an edge list, and the number of occurrences of H in G.
            - For example, a graph which consists of a single path of length 9 might have the following description: [[1,2], [2,3], [3,4], [4,5], [5,6], [6,7], [7,8], [8,9]]. If you counted 1213 instances of this graph in G, your file would contain a line of the form:

```
[[1,2], [2,3], [3,4], [4,5], [5,6], [6,7], [7,8], [8,9]] 1213
```

    - For each small graph whose instances you count, a list of all of its appearances in your data set. If you are able to get up to small graphs of a certain size, this list may become prohibitively long. In this case, for any small graph for which the list would be too large, please do not submit the list. But you should be prepared to produce/provide such a list in short order should we request it. The vertices in both H and G will be labeled by integers; each instance of H in G will be specified by listing the images of the vertices of H in G, in order.
        - For example, if H is a path of length 4 specified by the edge list [[0,1], [1,2], [2,3], [3,4]] and G has an induced path H’ that goes from 17 -> 24 -> 2 -> 107 -> 1117, then you would specify H’ as an instance of H by the list [17,24,2,107,1117]. (Note that the order doesn’t matter, since a subset of the vertices of G uniquely specifies an induced subgraph.)
3. **An annotated bibliography**. List all references for your project with a short paragraph for each (2-3 sentences) that describes how the work contributed to your project.
4. A **video presentation** of your project
    - mp4 or QuickTime format

    - 7 minutes or less.
    - This is a technical presentation to your professors in this class
    - You can assume a general familiarity with networks/graphs (say, at the level described in this document), algorithms, and asymptotic analysis, but you cannot assume knowledge of specific algorithms and you should refrain from using jargon.
    - Clearly and concisely explain your solution, your results, and future directions.
    - You can assume your audience has your source code and please refer to specific sections as you describe your solution.
    - Do not treat this presentation as a code review. Do not step through every line of your program.

Grading

- Subgraph counts: **max 40 points**
    - Up to **10 points** for counting subgraph in each network, depending on accuracy of the counts, as well as how large the subgraphs were that you were able to count in under an hour on a standard laptop
- Annotated bibliography: **max 10 points**
- Video presentation: **max 50 points**
    - Half of the points will depend on clarity and half on technical correctness
    - Solution, algorithm and code (suggest ~4-5 min): **max 40 point**s
        - a clear explanation of your solution: **max 20 points**
        - implement an algorithm from the literature or a non-trivial algorithm of your own design (i.e., better than testing all subsets of k vertices in G): **max 20 points**
    - Results (suggest ~1-2 min): **max 5 points**
    - Future directions (i.e., what would you do with more time?) (about 1 min): **max 5 points**
- Projects will not be graded unless all 4 components (1. source code, 2. subgraph counts for any of the 4 datasets you choose to analyze - and you must analyze at least one, 3. annotated bibliography, and 4. video) are received.

References

1. Milo et al., Science, 298(5594):824-827, 2002.